Check 41. The prediction of software complexity.pdf

by Arta Sundjaja

Submission date: 20-May-2019 11:31AM (UTC+0700)
Submission ID: 1102563461
File name: 41._The_prediction_of_software_complexity.pdf (306.26K)
Word count: 3090
Character count: 17405

The Prediction Of Software Complexity Based On Complexity Requirement Using Artificial Neural Network

Wartika Memed Purawinata[1,2]
[1] Faculty Of Engineering and Computer Science,
Indonesian Computer University,
Bandung, Indonesia 40132
[2] Computer Science Department,
BINUS Graduate Program - Doctor of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
Wartika@email.unikom.ac.id

Ford Lumban Gaol[1], Ariadi Nugroho[2]
Computer Science Department, BINUS Graduate Program - Doctor of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
Fgaol@binus.edu[1],
Ariadi.nugroho@binus.ac.id[2]

Bahtiar Saleh Abbas
Industrial Engineering Department
Faculty of Engineering, Bina Nusantara University,
Jakarta, Indonesia 11480
Bahtiars@binus.edu

***Abstract*—In the recent years, the productivity of software has grown in size, complexity, and also cost. As that software productivity growth , several problems has been appeared in software project management especially that correlated to complexity. One of complexity factors is requirement. A unit of requirement used as an option to the design phase of product development . The requirement is also a main option in verification process. So the the requirement complexity in this research is used as parameter to predict the software complexity. Because of the data pattern to connect between the requirement and the complexity is complex. So that this paper attempt to make a connectivity model between requirement complexity and prediction complexity of software using artificial neural network method with Levenberg Marquadt and Bayesian Regulation algorithm. So it can be seen comparison of experimental results by using the two algorithms.**

Keywords—prediction; complexity; requirement; software; artificial neural network

I. INTRODUCTION

Nowadays, the guarantee of quality, customers satisfaction and the reliable products are the urgent need of software industries [1]. The quality of software depends on several factors such as the accuracy of delivery time, within budget and meeting the users need. [2] in his research showed that the development of software project has high inaccuracy and low success. [3] Stated that there are six of main cause of software project failure such as an incomplete requirement, the low of user involvement, the lack of sources, the high of expectations, the lack of executive support, the changes of requirement and specification.

As the increasing implementation of information technology in various fields, so that the software complexity is also increasing. The increasing of that complexity will be related with several increasing of the requirement. [4] in his research stated that the complexity is the main support of cost, reliability, performance, and one of the most important factors that can affect software quality. In recent decades, software complexity has created a new era in computer science [5].

The best option for predicting software complexity is before software development reaches the coding phase [6] . This helps developers in managing software projects to assess software early in the process so that it can make changes to reduce complexity and improve the long-term utility of the product [7] .

The development research about software complexity in the early phases of life cycle has been widely practiced. [4] and [6] proposed a software-based complexity measurement requirement. In his research [4] and [6] stated that the measurement of software complexity is done by summing functional requirements, sub-functional, and non-functional requirements.

Because of the data patterns for connecting the requirements and complexity of this software complex. Therefore, this paper attempts to make a model of the relationship between the complexity of requirements and the prediction of software complexity using artificial neural network methods. Thus, it is expected to find out the result of the software complexity prediction based on the complexity of requirements by using artificial neural network method.

II. LITERATURE REVIEW

A. The Software Complexity

[8] Proposed that software complexity refers to software characteristics that affect the level of resources used by a person in performing a given task. According to [9] software complexity is the degree of difficulty in analyzing, designing, modifying, maintaining, and testing software.

B. *Requirement*

Requirement is the needs of physical and functional that is documented which must be designed, either in the form of a particular product or process. In this case it may be an identification of the attributes, abilities, characteristics, or system quality which necessary has any value and benefits for customers, organizations, internal users, or other stakeholders. IEEE defines the requirement as (1) a condition or capability required by a user to solve a problem or achieve a goal. (2) a condition or capability that must be met or owned by the system or component of the system to fulfill any other formally applicable contracts, standards, specifications or documents.

IEEE classifies the requirements into three categories: functional requirements, non-functional requirements, and domain requirements. In the classical approach technique, a set of requirements is used as input to the design stage of product development. Requirement is also an important input in the verification process, because testing should be traced back to a specific requirement. The requirements also indicate which elements and functions are required for a particular project. This is reflected in the waterfall model of the software life cycle. However, when iterative methods, the development of software is used, the system requirements is developed gradually simultaneously with the design and implementation.

The functional requirements can be a calculation, technical details, data manipulation and processing, and other specific functions that determine what the system should achieve (Keshavarz 2011). Functional requirement states the basic functions that the software system must perform in receiving and processing inputs and processing in generating output. Facilitating the development process usually the system is divided into a number of modules or smaller units called sub-function or sub-process.

Functional requirement states a basic function that must have done by the software system in accepting and processing of input and management for resulting the output. To facilitate the development process the system usually divides into several modul or smaller unit that called sub function or sub process. The equation to calculate the functional requirement according to [6] as follow:

$$FR = \left(\sum_{i=1}^{n} No.of\ Fn\ x\ No.of\ sub\ Fn \right) \quad (1)$$

The equation for calculating the functional requirement according to [10] is as follow:

$$FR = \left(\sum_{i=1}^{n} (Functionality)i \right) x \left(\sum_{j=1}^{3} (Sub\ process\ decomposition)ij \right) \quad (2)$$

Non function requirement refers to the qualitative needs of the system. Non functional requirement (also known as quality requirements) is associated with attributes system such as reliability and response time. Non functional requirement arises because of user needs, budget constraints, organizational policies, and so on. These requirements are not directly related to the specific functionality provided by the system. Non functional requirements should be done in software for efficient performance. Various types of non functional requirement as interoperability, implementation, standards, efficiency, reliability, portability, usability, etc.

Non Functional Requirement [11] can be calculated as an equation:

$$NFR = \left(\sum_{i=1}^{3} Coefficient\ i\ x\ No\ i \right) \quad (3)$$

No*i* is the number of non functional needs with the importance degree of type i and coefficient i is the importance coefficient of type i

Non *Functional Requirement* [10] :

$$NFR = \left(\sum_{i=1}^{n} (Attribute)i \right) x \left(\sum_{j=1}^{m} (Subattribute)ij \right) \quad (4)$$

NFR according to [6] is calculated as the equation:

$$NFR = \left(\sum_{i=1}^{3} \sum_{j=1}^{n} (Type\ i * count\ j) \right) \quad (5)$$

C. *Artificial Neural Network*

[12] Artificial neural network is a branch of artificial intelligence (artificial intelligence). The human's brain consists of 10 billion neurons that are interconnected with each other. These relationships are called synapses. Neurons are broadly divided into three namely, cell body, dendrites, and axons. Cell body functions to process the incoming signal, dendrites is the input unit as the entry point of the signal, axon is the output unit of the signal which is resulted from the cell body process. The relationship between neurons and other neurons through a synapse relationship.

The basic element of artificial neural network consists of 3 main parts, they are weight, threshold, and activation function.

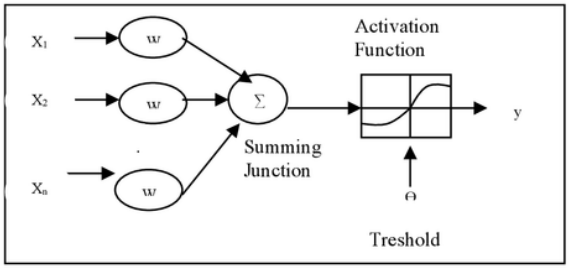

Fig. 1. The elements of artificial neural network

- Input: X1, X2, X3, ... Xn, is a signal that enter the nerve cell.
- Weight : W1, W2, W3, ... Wn is a weight factors associated with each node. Each input will be multiplied by

the weight of its respective node, $X^T.W$. Depending on the activation function used, $X^T.W$ values can excite nodes or inhibit nodes

- Threshold: the internal threshold value of the node Θ is the magnitude of the offset affecting the activation of the output node y:

$$y = \sum_{i=1}^{n} XiWi - \Theta \quad (6)$$

- Activation function: is a mathematical operation applied to the output signal y.

$$y = f(X_1W_1 + X_2W_2 + \ldots + X_nW_n) \text{ or } y = f(x.w) \quad (7)$$

Some of the activation functions commonly used in artificial neural networks include:

1) Linier function

Linear function is expressed by the equation:

$$y = f(x) = \alpha x \quad (8)$$

Where α is the slope of the function. If α = 1 then the activation function is an identity function

2) Threshold (hard-limiter) function

Function of threshold type: binary and bipolar. The binary threshold function has the y output:

$$y = f(x) = 0 \text{ if } x < 0, 1 \text{ if } x \geq 0 \quad (9)$$

The bipolar threshold function has output y:

$$y = f(x) = -1 \text{ if } x < 0, +1 \text{ if } x \geq 0 \quad (10)$$

3) Linear piecewise function

The mathematical equations of this function are:

$$y = f(x) = -1 \text{ if } x < -1, x \text{ if } -1 \leq x \leq 1, 1 \text{ if } x \geq 0 \quad (11)$$

4) Biner sigmoid function

It is the most common non-linear function used in artificial neural networks. The sigmoid activation function can be written as follows:

$$y = f(x) = \frac{1}{1+e^{-\alpha x}} \text{ for } 0 \leq f(x) \leq 1 \quad (12)$$

α is the shape parameter of the sigmoid function. By changing the value of α then the shape of the sigmoid function will vary.

5) Bipolar sigmoid and tangenhiperbolik (tanh) function

This function is expressed by the following equation:

$$y = f(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}} \text{ untuk } -1 \leq f(x) \leq 1 \quad (13)$$

α is the shape parameter of the tanh function. By changing the price α then the shape of the tanh function will vary.

- Bias and threshold

The most basic architecture of artificial neural networks is a single-layer artificial neural network: it consists of several input units and one output unit. Usually in the input unit plus a variable that is bias (b) or threshold (Θ).

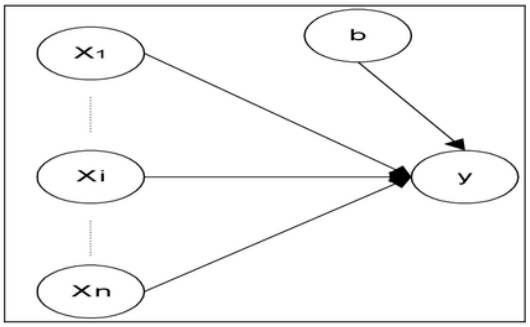

Fig. 2. Artificial neural networks with bias

If we use bias b then the value of the activation function will be as follow:

$$y = f(net) = 1 \text{ jika net} \geq \Theta, -1 \text{ jika net} < \Theta \quad (14)$$
$$\text{where net} = \sum xiwi$$

D. Basic Principles Of Artificial Neural Network Training

Artificial neural networks have the ability to learn almost the same as the nerves in humans. [13] Classifies artificial neural networks into two, that are, how artificial neural networks store knowledge / encode (learning process) and how artificial neural networks process and respond incoming / decoded data. [12]

- The encoding process consists of 2, they are (1) supervised (networked, the network is input and the output is determined by the teacher.) During the learning process the network will adjust the synaptic weight. (2) Unsupervised, the network is fed and the output will be regulated independently according to the rules they have.

- The decode process consists of 2: (1) Feedforward (without feedback) and (2) feedback.
- A group of vector pairs and the desire output vector (target) is called training pattern
- The pairs of input vector and target embedded together with weight value of connection (synaptic) into network. This process is done in learning/practice process
- Learning process : (1) the network is given an input and also intended output pairs (target), (2)the network is made any calculation to input data and resulting a temporarily output, (3) comparing between temporary output and target output and the difference is used for fixing the synaptic weigh value, (4) that process will redo untill the error or the difference of temporary output and the smaller target or convergent.
- The learning process will be finished if the matrix value of connection weigh that resulting can make a system which can give the perfect pattern even though the input pattern that is given to the network is not complete or it is affected by noise

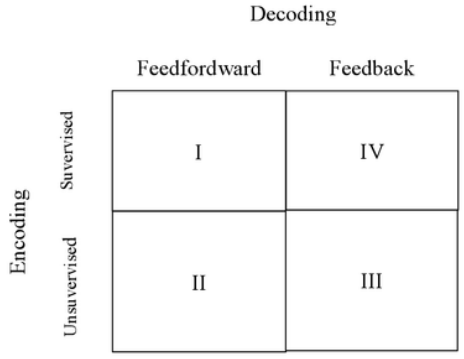

Fig. 3. Artificial neural network clasification

- Kwadran I : supervised – feedforward

  The learning process (learning repository) is guided and in the process of responding the input data then does not provide any feedback.
- Kwadran II : unsupervised – feedfordward

  In learning process (learning repository) is not guided and in input data process does not provide a feedback.
- Kwadran III : unsupervised – feedback

  In learning process (learning repository) is not guided and in responding input process provide a feedback.
- Kwadran IV : supervised – feedback

  In learning process (learning repository) is guided and in responding input process provide a feedback
- There are several artificial neuron network models correspond to the neuron configuration and its algorithm, such as : (1) Hebb, (2) Hopfield, (3) Hamming, (4) Kohonen, (5) Levenberg – Marquadt, (6) Bayesian Regulation, etc.

IV. Discussion

In this discussion will be described the experimental research design, data used, experiments, analysis and conclusions.

A. Research Design

Research design in this research is used a, experiment research design. The research steps can be described as follows: (1) Determination of hypothesis that the complexity of the requirement affects the prediction of software complexity, (2) Selecting experimental unit in this data case for training using data set of UCI machine learning repository, (3) Experimental design, (4) Experiments, (5) Analysis, (6) Conclusion then do experiments, and experimental results are analyzed for conclusions.

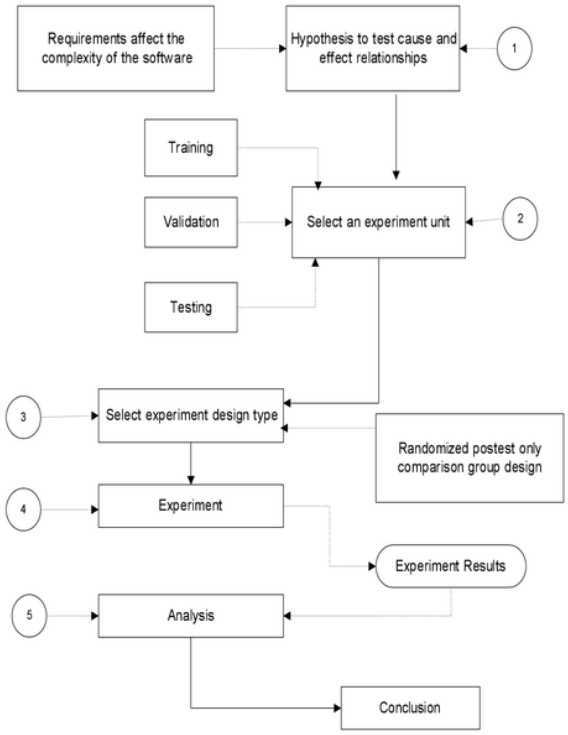

Fig. 4. Experiment design diagram

B. The applied attributes

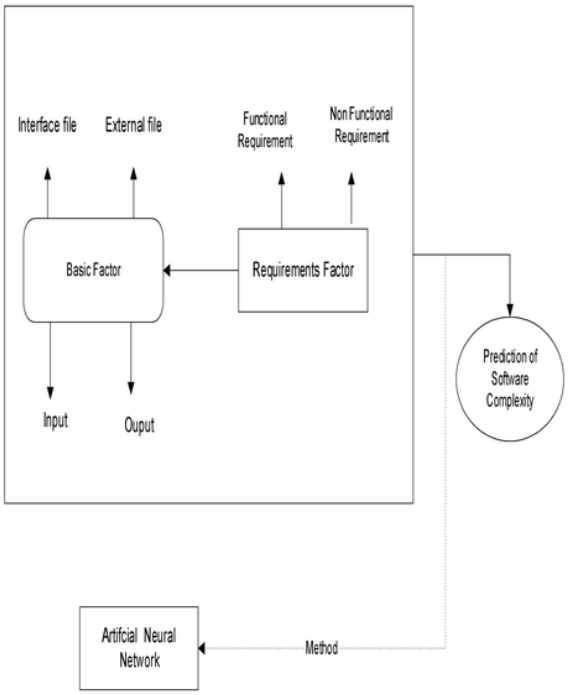

fig. 5. The applied attributes

The attributes used in this research is described in fig 5 above where the attribute is used to predict software complexity is a basic factor, functional requirement and non functional requirement.

C. The Experiment Design

The experiment design type is used randomized posttest only comparison group design. In this design, there were randomization procedures of the subjects in the group, then comparing, using two or more variations of the independent variables and dividing in two or more groups, the treatment was performed on all groups.

D. Experiment

The experiment input consists of three atributes, they are basic factor, functional requiremens, and non functional requirements. The software complexity is calculated based on the requirement complexity. Method used artificial neural network with algorithm levenberg Marquadt and Bayesian Regulation. The following mathematical equations were adopted from the study [6] and [14]:

Software Complexity = Basic Factor + Functional Requirement + Non Functiopnal Requirement (15)

TABLE 1. EXPERIMENT RESULTS

Algorithm	Eksperimen 1	Eksperimen2
Data division	Random	Random
Training	Levenberg – Marquadt	Bayesian Regulation
Performance	Mean Squared Error	Mean Squared Error
Progress :		
Epoch	1000 iteration	1000 iteration
Time	0:10:02	0:19:35
Performance	9.61E-08	2.10E-11
Gradient	8.90E-05	8.31E-10
MU	1.00E-05	5.00E+06
Validation Check	0	0
Effective # Param		23.8
Sum Squared Param		18.8
Number of Hidden layer	5	5

The data division is that portion of the program set aside to identify explicitly all the format characteristics of data received by the program, created within the program, and produced as output. In this study data division is random. for training in experiment 1 used the classification algorithm Levenberg – Marquadt, and for training in experiment 2 used the classification algorithm Bayesian Regulation. To evaluate performance use mean squared error. Man squared error is a measure of the quality of an estimator, it is always non-negative, and values closer to zero are better. Epoch is a step done in learning on ANN. If the magnitude of the epoch is greater than that specified, then the learning process will stop. The gradient or slope of a line is the magnitude of the angle formed by the line against the horizontal line. The magnitude of the slope starts from the negativity up to and up to infinity. The hidden layer consists of neurons that receive data from the input layer.

TABLE 2. MSE EXPERIMENT

		Levenberg – Marquadt		Bayesian Regulation	
	Samples	MSE	R	MSE	R
Training	574	960764e-8	9.9999e-1	2.01489e-10	9.9999e-1
Validation	123	4.99140e-8	9.9999e-1	0.0000e-0	0.0000e-0
Testing	123	7.92656e-8	9.9999e-1	2.52823e-11	9.9999e-1

Table 2 shows comparison of the mean squared error from the experimental results between Levenberg – Marquadt algorithm and Bayesian Regulation algorithm on training, validation and testing. Training set using 574 samples, validation set using 123 samples, and testing set using 123 sampelas.

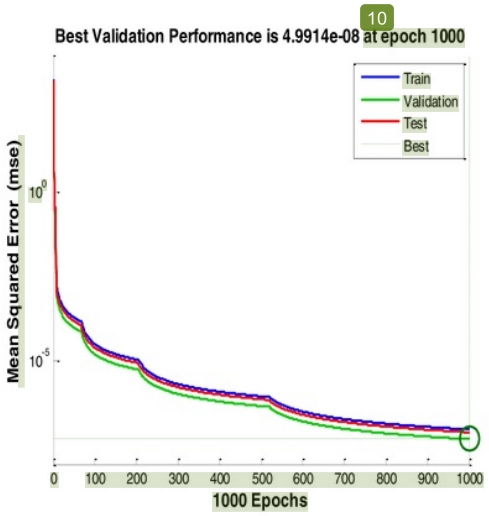

Fig. 6. Performance Levenberg – Marquadt algorithm

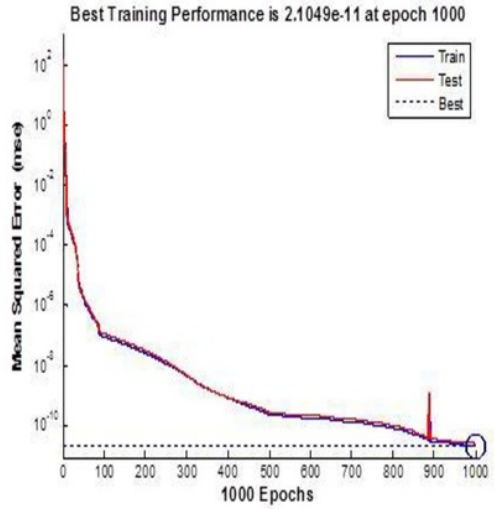

Fig. 7. Performance Bayesian Regulation algorithm

V. CONCLUSION

From the experiment results that have been done using three attributes, five hidden layer, artificial neural network method with Levenberg - Marquadt and bayesian regulation. Performance is based on Mean Squared Error. It can be concluded that the prediction of software complexity based on complexity requirements using the Levenberg-Marquadt algorithm is better because resulted smaller Mean Square Error.

REFERENCES

[1] X. Liu, "Object-oriented software metrics," 1999.

[2] K. Na, J. T. Simpson, X. Li, T. Singh, and K. Kim, "Software development risk and project performance measurement : Evidence in Korea," vol. 80, pp. 596–605, 2007.

[3] B. Boehm, C. Abts, and A. Brown, "Cost estimation with COCOMO II," *ed: Upper Saddle …*, 2000.

[4] G. Keshavarz, N. Modiri, and M. Pedram, "Metric for Early Measurement of Software Complexity," *Interfaces*, 2011.

[5] A. West, "NASA study on flight software complexity," *Final Report. Technical report, NASA*, 2009.

[6] B. Arthi, A. G. Selvarani, and A. Sathya, "Software Complexity Measure from Software Requirements Document and Cost Driver Factors," vol. 24, pp. 6–12, 2016.

[7] R. Pressman, "Software engineering: a practitioner's approach," 2005.

[8] B. Sellers, "Ojbect-oriented metrics. measures of complexity," 1996.

[9] S. Nystedt and C. Sandros, "Software Complexity and Project Performance," *University of Gothenburg*, 1999.

[10] A. Sharma and D. Singh, "Estimation of Software Development Effort from Requirements Based Complexity," *Procedia Technology*, vol. 4, pp. 716–722, 2012.

[11] G. Keshavarz, "Metric for Early Measurement of Software Complexity," vol. 3, no. 6, pp. 2482–2490, 2011.

[12] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*. 2005.

[13] L. Zadeh, "Soft computing and fuzzy logic," *IEEE software*, 1994.

[14] A. Sharma and D. S. Kushwaha, "A Complexity measure based on Requirement Engineering Document," vol. 1, no. 1, pp. 112–117, 2010.

[15] Warnars, H.L.H.S. 2011. Object Oriented Modeling with unified modelling language 2.0 for simple Software Application based on Agile methodology. Behaviour & Information Technology an International Journal, 30(3), 293-307.

Check 41. The prediction of software complexity.pdf

ORIGINALITY REPORT

20%	10%	9%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.net Internet Source	3%
2	Submitted to Universiti Teknologi Malaysia Student Paper	3%
3	Submitted to Vishwashanti Gurukul Student Paper	2%
4	Winda Astuti, Satrio Dewanto, Khristian Edi Nugroho Soebandrija, Sofyan Tan. "Automatic fruit classification using support vector machines: a comparison with artificial neural network", IOP Conference Series: Earth and Environmental Science, 2018 Publication	2%
5	www.idosi.org Internet Source	2%
6	www.enggjournals.com Internet Source	2%
7	Zilouchian, Ali. "Fundamentals of Neural Networks", Intelligent Control Systems Using	1%

Soft Computing Methodologies, 2001.
Publication

8	Evawaty Tanuar, Bahtiar Saleh Abbas, Agung Trisetyarso, Chul-Ho Kang, Ford Lumban Gaol, Wayan Suparta. "Back propagation neural network experiment on team matchmaking MOBA game", 2018 International Conference on Information and Communications Technology (ICOIACT), 2018 Publication	1%
9	Submitted to Prague College Student Paper	1%
10	Submitted to Netaji Subhas Institute of Technology Student Paper	1%
11	Submitted to Po Leung Kuk Choi Kai Yau School Student Paper	1%
12	7thsastech.khi.ac.ir Internet Source	1%
13	Submitted to NCC Education Student Paper	1%
14	Submitted to Florida State University Student Paper	1%
15	Suryadiputra Liawatimena, Harco Leslie Hendric Spits Warnars, Agung Trisetyarso, Edi	1%

Abdurahman et al. "Django Web Framework Software Metrics Measurement Using Radon and Pylint", 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), 2018

Publication

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		